type. If the filter a delete type (i.e., MRPIs that are filtered out are deleted instead of not being displayed), the MRPI and its children MRPIs are immediately deleted in step 179 so no other filter need process them. Any MRPI that has any parent MRPI that is excluded is itself excluded. If the filter is not a delete type, then the flag is set to exclude in step 180 for this filter in the current MRPI's MRPF filter affects list. Any MRPI that is excluded by any active filter is not displayed in the currently open DDs 20. Next, step 181 determines whether there are additional filters to be processed. If so the method loops to 171, otherwise the method tests whether this is the last MRPI to be processed in step 182. If there are more MRPIs to process, the method returns to step 170, otherwise, the process is complete.

As has been noted above, Dynamic Documents are used to display XML document data to the user. Essentially, DDs are a rearranged and reformatted MRT. A DD is a window on the display device 14 comprised of a hierarchy of display panes. Each pane displays one or more of the MRT's PRTs and calculated fields. The user can define functions, sorts, and filters for each DD. A DD is also a tool for users to update a PRI and thereby the MRs and DDs that reference the PRT. Each DD corresponds to one MRT and each DD pane corresponds to one or more of the PRTs used by its MRT. The advantage of using DDs to create multiple data view and update transactions is that the data relationships can be radically restructured from that of the MRT. Child records can become parents, records can share fields, calculated fields can be created, and sets of records can be summarized into single records and values.

The DDs are created by the user based on the types of information that the user wants to view and update. The user selects which of the PRTs in the MRT to include in the dynamic document pointer (DDP). For user viewing or update, the invention creates DDP records by accessing the field values in its underlying PRT's PRIs and its underlying MRPI's calculated fields. Any PRT in the MRT can be used in the DDP and can be used more than once in a single DD. While their shared hook predefines the relationship between any two PRTs in a MRT, the user defines the relationship between any two PRTs in a DD. By putting a PRT in a parent hierarchy pane, the user is making that PRT the parent of all the PRTs in the children panes even if the PRTs have a different relationship in the MRT. This capability allows the user to represent the data in virtually anyway they want even though that representation is not supported by the production data

source in which the data is stored. For example, a customer order MRT that has its Items PRT hooked to its Order Lines PRT which in turn is hooked to its Order Header PRT and also has Salespersons separately hooked to its Order Header PRT, as described above with reference to FIG. 3, can have a DD where the Item PRT is in the top level pane and the Salesperson is in a child pane. In this example, the Item has become the parent of Salesperson. The DD displays in the top pane any Item used in any Customer Order MRT, and displays in the child pane all the Salespersons that sold that Item on any Customer Order.

When the user places a PRT in a DDP, the user gains access not only to the PRT's fields in that pane and any of its child panes, but also access to all the calculated field of the PRT's MRPF. These calculated fields are comprised of fields of the selected PRT and also any field from any other PRT in the MRT. In the above example of the Items pane, the user could display the total amount ordered on all orders for each item even though the DD does not include the order lines or order header PRTs which are required for this calculation.

The user creates the DD Pane definition by entering the pane's name, its position in the pane hierarchy and then by selecting as many of the PRTs of the MRT that are to appear in the pane. The system then automatically joins the PRT's MRPIs together to create the DDPs. These DDPs are the DD equivalent of MRT's MRPIs. Each DDP corresponds to a single pane display row and can roll up into display subtotal rows.

The user specifies how to update production data sources (other than the PRTs) by defining as many separated transactions types as desired. Then the system, for any single update to the DD, automatically creates Production Data source Update Transactions Instances (UTs) and posts them to the appropriate data source. Each Production Data Source Update Transaction Type (PDUTT) can update as many PRTs as the user wishes; the only restriction is that they all be in the same instance of a production data source. A DD's PDUTTs are similar to its pane attributes and report attributes in that it formats the field into an output medium. One primary difference is that a PDUTT does not always request the current value; it can also get the amount a value has changed by as a result of a transaction, or a literal amount stored in the PDUTT. Also, a PDUTT is different in that its instances (UTs) are only created when the DDs underlying MRIs are changed. The user can have all MRI changes result in UTs or can limit them to only changes entered through the DD

of the PDUTT, or only when the DD is active (i.e. opened by the user).

Referring now to FIGS. 9A and 9B, the preferred method for creating DDPs is described. First, the DD 20 is read in step 190. Then the highest hierarchy level pane definition not yet processed is read in step 191. In step 192, the pane's PRT and MRPF are read. Next, the method tests whether the MRPF is already used in an existing pane in step 193. If so, the method jumps to step 196 and avoids having to initialize the pane. On the other hand, if the pane does not exist, it must be created. The method creates the pane in steps 194 and 195. In step 194, the method runs the pre-join filters on the MRPF's MRPIs. In step 195, the method joins the MRPIs to previously selected MRPI's for this pane, and the one from the parent pane. The panes are preferably joined on MRT identification and parent MRPF ID fields. The method then continues in step 196. In step 196, the method determines if this is the last PRT to be processed. If not, the process returns to step 192 and reads other PRTs. Otherwise, the method continues in step 197. In step 197, the method runs all post join filters. Then in step 198, the method tests whether any child panes for this PRT exist. If there are child panes, the method saves a copy of the joined MRPI for each child pane and continues directly to step 200. If there are no child panes the method proceeds directly to step 200. In step 200, the method determines if this pane or its parent panes include lead PRT for all its MRPFs. If lead PRTs are included, the method executes steps 201 and 202. In step 201, the system summarized the joined MRPI's by identification of the PRTs in this and the parent panes. If all the panes include lead PRTs for each MRPF, the summarization step 201 would have no effect; that is why it is shipped. The system in step 202 then runs all post summarize filters, and moves to step 203. If lead pointers are not included, the method moves directly to step 203 and determines whether this is the last pane to modify. If not, the method loops to step 191, otherwise the process ends.

FIG. 10 illustrates the preferred method for updating a MRT for a new PRI. The method starts by reading the new PRI in step 210. Then in step 211, the PRT definition and the where used list of this PRI are read. Next in step 212, the definition of the MRT listed in the where used list of step 211 is read. Next, the method determines if the PRT is a lead PRT of a MRPF in step 213. If it is a lead PRT, the method continues in step 214 where a MRPI is created. Then in step 215, the function and filters for the MRPI are executed. If it determined not to be a lead PRT in step 213,